Accelerating Machine Learning on Chameleon Cloud

Arun Das

University of Texas at San Antonio

arun.das@my.utsa.edu
linkedin.com/in/arun-das

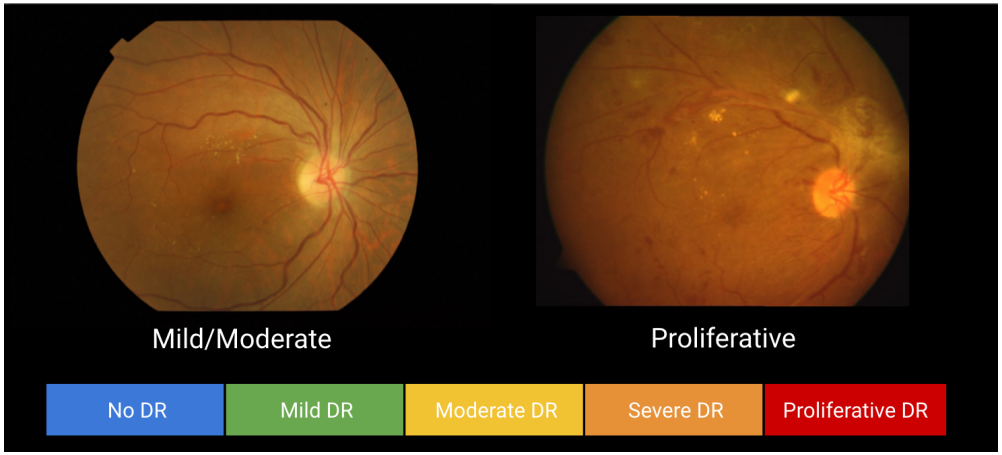
September 13, 2017

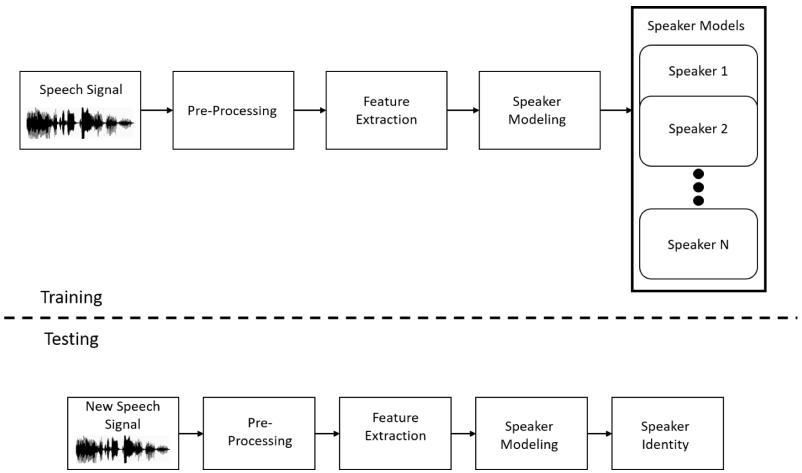
Overview

What is Machine Learning?

Coined in 1959 by Arthur Samuel, a pioneer in AI, Machine Learning is a subfield of Computer Science which gives computers 'the ability to learn without being explicitly programmed'.

What is Machine Learning?





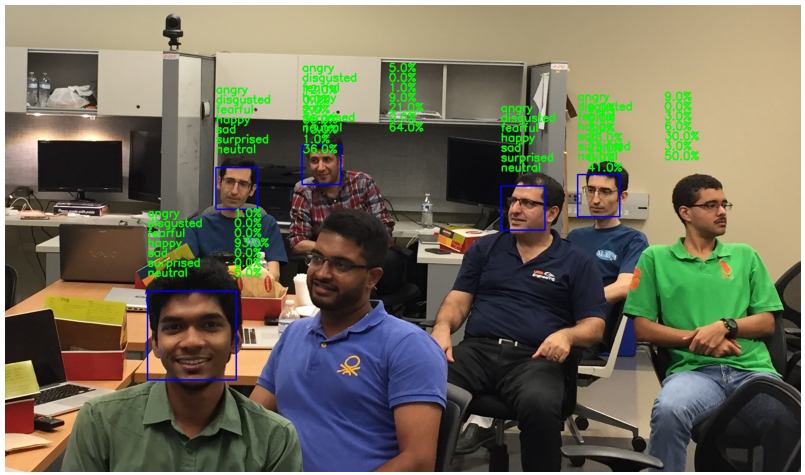


Figure: Advances in Machine Learning, Deep Learning and Artificial Intelligence [SS, DR].

Why Machine Learning?

- With the advancement of Artificial Intelligence (AI) and Deep Learning (DL) to bring once sci-fi solutions to reality, human race is in an imperative shift towards autonomous, reliable and scalable solutions to critical problems.
- As discussed previously, these problems span from natural language processing, computer vision to health care, even assisting doctors make informed decisions quicker and more efficiently with help of AI enabled applications trained to do specific tasks.
- Crunch huge amount of data and analyze them, often at higher levels of accuracy than humans, realtime.
- Enable different levels of automation to various domains such as security, decision making, resource utilization, etc.

The big picture

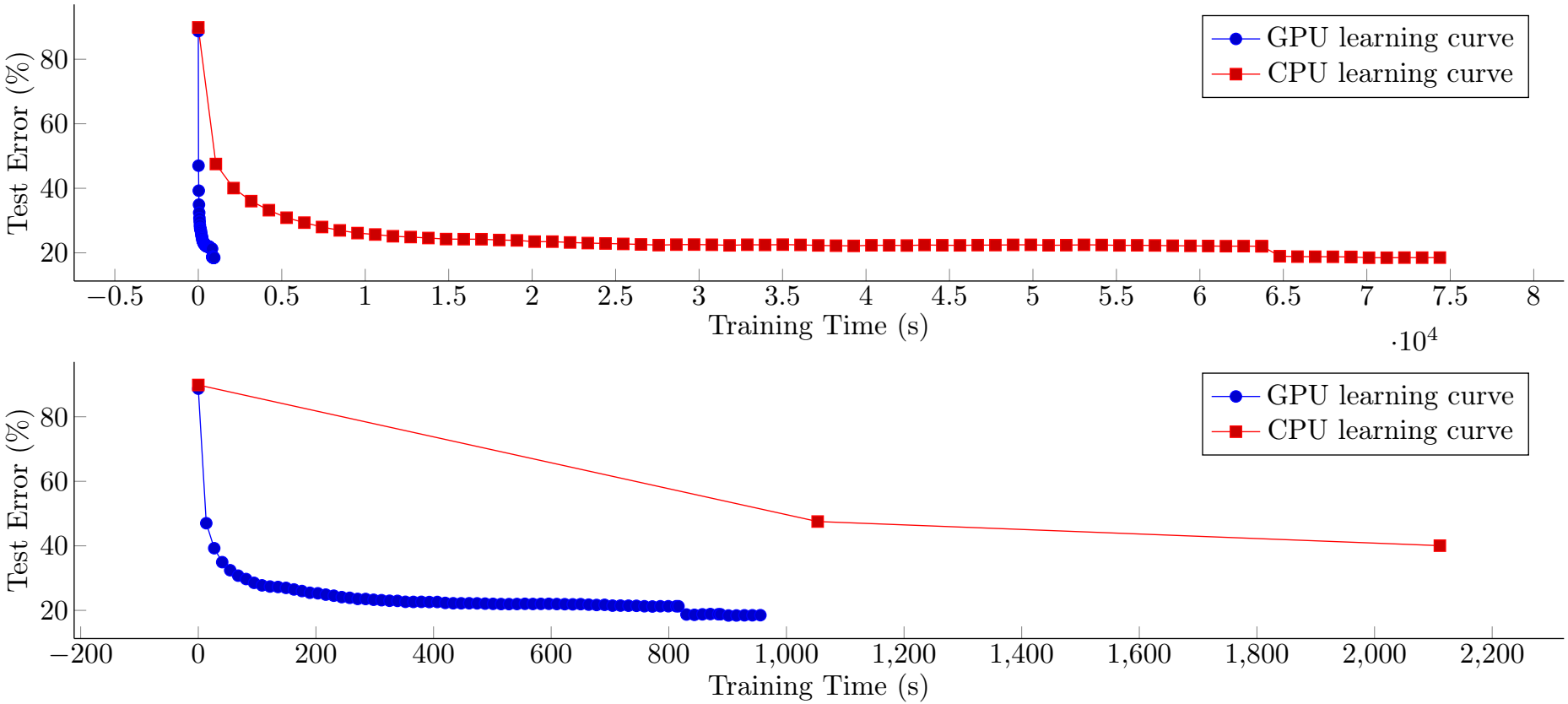
- We live in an age where computer systems mimics and/or replicates human intelligence, the age of Artificial Intelligence (AI).
- Neural Networks are used extensively to train AI models for carrying out everyday human tasks, which we do involuntarily (for example, understanding objects, animals, etc.).
- Deep Learning, a subset of Machine Learning, which uses neural networks to learn tasks is driving AI research at a rapid phase.
- More and more academic papers being published around AI, industry embracing AI.

Why Now, ML was around for a long time?

- The big-data explosion and easier access to compute helped accelerate deep-learning research thereby attracted more researchers to AI.
- In the age of computers where Amdhal's law struggle to be relevant, AI researchers use Graphics Processing Units (GPUs) to parallelize individual computations that occur in DL based neural networks.
- For DL networks such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), multiple folds of speedup can be achieved by using desktop grade GPUs for parallelizing the mathematical operations.
- Convolution operations are, mathematically, almost 80% matrix multiplications. These matrix multiplications can be effectively and efficiently parallelized on GPUs by using low-level kernel libraries such as NVIDIA cuDNN (CUDA Neural Network) library.

Why GPUs?

Training a neural network for 10 class image classification on CIFAR-10 dataset, 60 million parameters and 650,000 neurons, 5 convolutional layers.



20 hours vs. 16 minutes, wait what?

Why one GPU is not enough?

- As researchers focus on tougher and bigger problems such as natural language translation, speech recognition, etc., often surpassing the state-of-the-art, the computational complexity the algorithms put forward is extraordinary.
- 125+ hours to train Inception v3 on Chameleon M40 GPU node.
- 250+ hours to train an Image Captioning model on Chameleon M40 GPU node.
- To cope up with the computation hungry learning algorithms, researchers use a cluster of powerful compute nodes to train neural networks.
- This distributed way of teaching neural networks to learn representations is often called Distributed Deep Learning.

Accelerating ML on Chameleon Cloud

- The power hungry deep learning algorithms need an efficient way of distributing its workload. This can be done in different ways:
- Workload can be distributed by adding multiple GPUs in the same machine.
- Explore parallelism across multiple machines.

Distributed Deep Learning

Data Parallelism

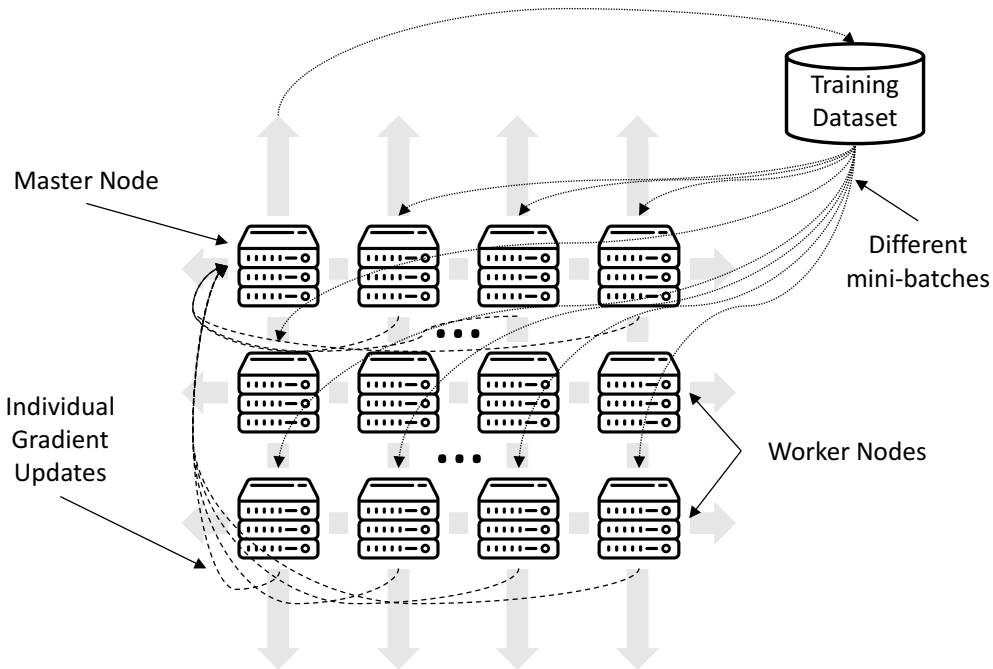
Data parallel models make use of small mini-batches to parallelize computations on different machines. It is assumed that a copy of the neural network model to be trained is available in all the worker nodes. The distributed deep learning algorithm schedules operations on different machines on small batches of training data and synchronizes the weight updates across them.

Model Parallelism

Model parallel models are used when one neural network model does not fit in the machine available. This can happen if the network is very deep, for example CNNs of hundreds of layers or dense and deep RNNs. The model is carefully split into smaller sections (often layer by layer) and are distributed across multiple GPUs in the same machine. Each GPU thus gets a part of the parameters and computations associated with the huge neural network.

Data Parallelism

If there are n machines, each machine will get $1/n$ of the training data locally. If b is the batch size supported by each machine, the effective batch size for n machines will be $n * b$. It significantly increases the amount of data that can be processed per second, thereby speeding up the training procedure.



Distributed Deep Learning: Impact of Batch Size

Batch Size	AlexNet	Inception V3	ResNet 150
1	1067.86 (+- 4.86)	19.06 (+- 0.57)	10.15 (+- 0.17)
2	1442.78 (+- 4.73)	34.19 (+- 0.67)	18.92 (+- 0.13)
4	1861.16 (+- 3.63)	56.37 (+- 0.54)	33.11 (+- 0.20)
8	2168.15 (+- 3.23)	81.78 (+- 0.69)	48.74 (+- 0.39)
16	2296.58 (+- 8.13)	108.00 (+- 0.63)	65.57 (+- 0.10)
32	1960.96 (+- 1.17)	125.26 (+- 0.67)	77.46 (+- 0.24)

Table: Impact of choosing the right batch size. For AlexNet, batch size *= 16. Each row shows the images the CNN can crunch per second (images/sec).

Distributed Deep Learning: Impact of Multiple Machines

GPU instances	Images/second	Speedup
1	194.86	1X
2	389.81	2X
8	1466.31	7.5X

Table: Impact of distributing the workload, benchmarked for ResNet 50 CNN architecture on synthetic data using Apache MXNet, SSH connection between nodes, 1GPU per node.

Image Name	Type	Status	Public
das_arun_Ubuntu16_MXNET_DIST_Sept2017	Image	Active	Yes

Deploying ML on Chameleon

Accelerating ML on Chameleon: Roadblocks

- Poor performance for multi-GPU setup within each node. Example: Add NVIDIA NVLink capabilities to P100 GPUs to extract much more performance from them during distributed deep learning.
- Very low local storage (200GB) in baremetal nodes. Datasets used for DL are much bigger and don't fit in the baremetal servers.
- NFS is slow for DL, also cumbersome to setup and worries about lease expiration. Trouble keeping up with the GPUs.
- CloudFuse and other packages do not solve 'fetch data directly from Swift Object Storage'. Limitation of 10,000 files per folder. DL datasets are huge!
- Need longer Leases. Most of our DL models need to be saved before the network is optimized. Retraining never give consistent, reproducible results.

Accelerating ML on Chameleon: Suggestions

- More local storage in compute nodes, 1TB would be great.
- Faster storage options, such as SSD in GPU nodes.
- Complex Appliances for DL. Happy to collaborate.
- Multiple GPUs in the same machine. For example, 4 M40's or 8 K80s's in a single node, another 'node type'.

References

e-Lab Segmentation Demo (2016)

e-Lab Segmentation Demo

Available at https://www.youtube.com/watch?v=FGzodQt7y4Q

Google Brain

Diabetic Retinopathy Classification using Deep Neural Networks

Available at
https://research.google.com/teams/brain/healthcare/images/retina.png

The End

Appendix A

Things I checked out or at least looked into:

1. Fuse, to mount Swift into compute nodes.
2. CloudFuse
3. Swift Virtual File System
4. pycloudfuse
5. Hadoop HDFS

Problems I faced for the above :

1. Complied fuse properly. Wasn't able to run fuse / failed to understand docs to run fuse.
2. CloudFuse - 10000 objects per container restriction.
3. Swift Virtual File System - Does not work for a multi project, multi user Swift environment as of now.
4. pycloudfuse - 10000 objects per container restriction.
5. Hadoop FS - incompatibility with traditional deep learning platforms. Only works with deep learning platforms involving hadoop/swift.

Appendix B

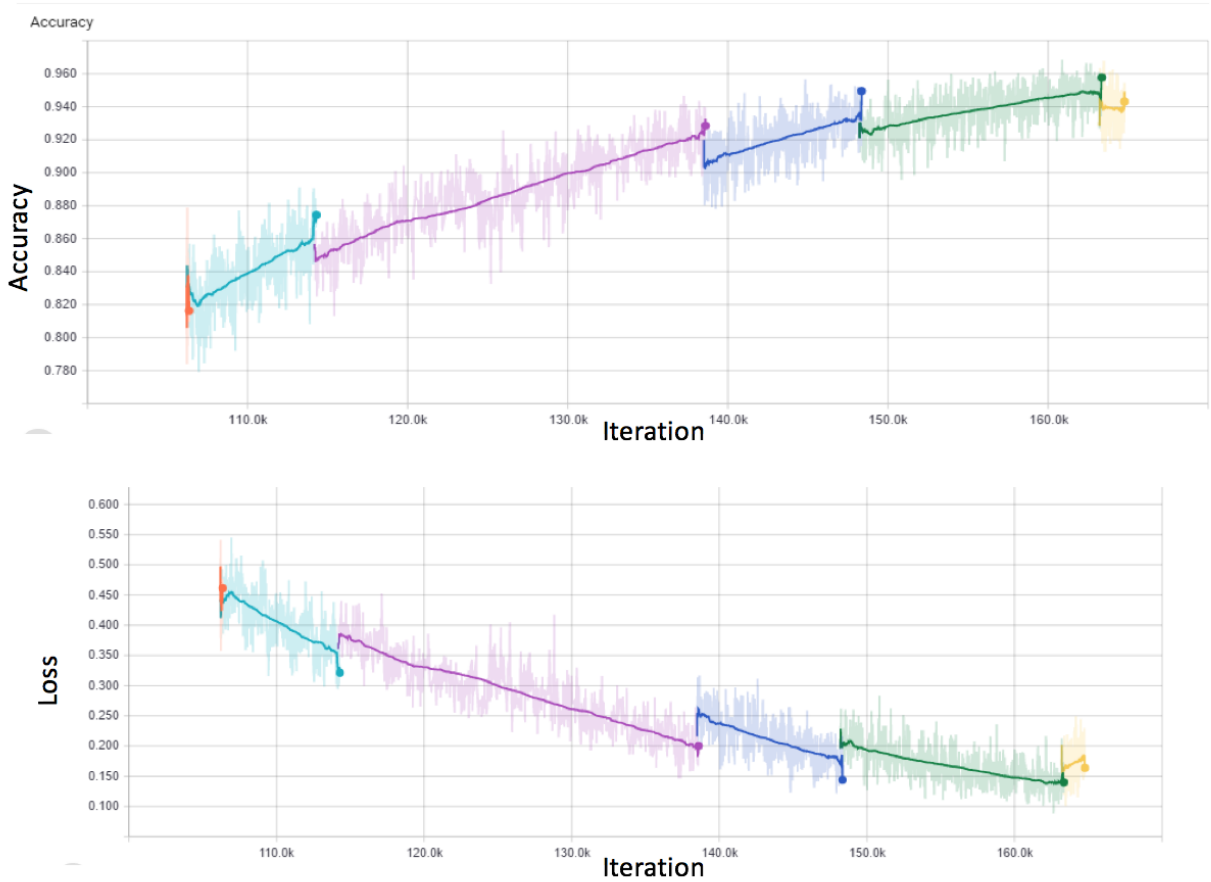


Figure: Save - Restore - Retrain a neural network. Unusable, unpublishable graphs! We need continuous runs.

Appendix C

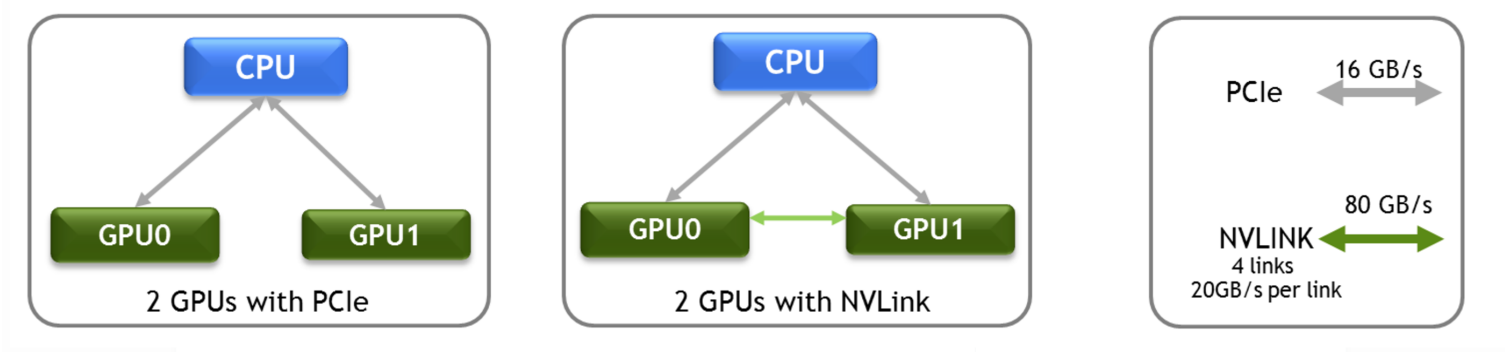


Figure: NVIDIA NVLink: Let's GPUs communicate much faster than PCI-e